The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume L-4/W1-2026
Free and Open Source Software for Geospatial (FOSS4G) 2026 – Academic Track "Bridging Geospatial Technology and Humanity",
30 August–5 September 2026, Hiroshima, Japan

Open-Source Spatiotemporal Traffic Congestion Analysis at City Scale: A Reproducible Python Pipeline Combining PySAL, OSMnx, and Multilevel Modeling

Firman Hadi[1], Fauzan Abdullah[2], Yasser Wahyuddin[1], L.M. Sabri[1]

[1] Department of Geodetic Engineering, Faculty of Engineering, Universitas Diponegoro, Semarang, Indonesia - firmanhadi21@lecturer.undip.ac.id
[2] Department of Statistics, Faculty of Sciences and Mathematics, Universitas Diponegoro, Semarang, Indonesia

Keywords: FOSS4G, reproducible pipeline, traffic analysis, PySAL, OSMnx, multilevel modelling

Abstract

City-scale spatiotemporal traffic analysis requires a long workflow—raw probe-data ingestion, segment matching to a stable reference network, temporal aggregation, exploratory spatial statistics, network-topology analysis, multilevel variance decomposition, and reproducible figure generation. Existing FOSS4G tools cover individual stages well (PySAL for spatial statistics, OSMnx for street networks, statsmodels for mixed models), but no integrated FOSS4G stack delivers all stages end-to-end with stable identifiers across provider data revisions. We present `traffic-congestion-pipeline`, an MIT-licensed Python package distributed on PyPI that closes this gap. The contribution comprises four engineering elements: (i) a provider-agnostic collector with HERE, TomTom, and Google back-ends behind a single interface; (ii) an OSM-based two-stage spatial matcher (geometric intersection followed by nearest-neighbour fallback) producing a stable `osm_composite_id` with 99.8% match rates across 50,000+ collection snapshots; (iii) a modular eleven-command CLI and Python API that allow each analysis stage to be run independently or chained into a single reproducible workflow; and (iv) demonstration on 316 million traffic observations from three Indonesian cities (Jakarta, Bandung, Semarang), processed end-to-end on consumer hardware in approximately fifteen minutes. The pipeline has supported a peer-reviewed empirical study and is, to our knowledge, the first FOSS4G stack to integrate provider-agnostic collection, OSM-stable segment identity, and multilevel variance decomposition into a single installable artifact.

1. Introduction

Spatiotemporal traffic analysis at city scale typically combines several distinct computational stages: continuous probe-data collection from a commercial API, alignment of provider-specific segment identifiers to a stable reference network, temporal aggregation to analytical periods, exploratory spatial statistics, network-topology metrics, and statistical modelling that respects the panel structure of the data. The Free and Open Source Software for Geospatial (FOSS4G) ecosystem contains mature components for every individual stage—PySAL (Rey and Anselin, 2007) for spatial autocorrelation, OSMnx (Boeing, 2017) for street-network graphs, `statsmodels` (Seabold and Perktold, 2010) for mixed-effects models, GeoPandas for geometry handling—but stitching them into a reproducible end-to-end workflow is left to each research group.

This integration gap matters in practice for three reasons. First, raw traffic probe data arrives as a stream of geometry-keyed snapshots whose internal identifiers drift between provider revisions; without a stable cross-snapshot identifier, longitudinal analysis is unreliable. Second, the analytical stages depend on each other: betweenness centrality computed on OSM nodes must be joined to the aggregated traffic panel before multilevel models can include it as a Level-2 predictor. Third, traffic researchers in resource-constrained settings are often the audience least equipped to assemble such a pipeline from scratch, yet they have the most to gain from open-source tooling. A turnkey integrated stack is needed.

Our contribution is `traffic-congestion-pipeline`, a Python package that closes this integration gap. It is the engineering counterpart to a parallel empirical study (cited below) which uses the pipeline to analyse 316 million traffic observations from three Indonesian cities; the empirical findings are reported there. This paper presents the *tool*: its architecture, its design decisions, its public APIs, and the trade-offs we encountered. The four specific contributions are:

1. **Provider-agnostic collector.** A pluggable collector with HERE, TomTom, and Google back-ends behind a single Python interface, enabling cross-provider experiments and provider migration without rewriting downstream code.
2. **OSM-based stable segment identity.** A two-stage spatial matcher that maps geometry-keyed provider snapshots to a persistent identifier derived from the OpenStreetMap reference network, achieving 99.8% match rates and median nearest-neighbour distance under 1.5 m across 50,000+ snapshots.
3. **Modular eleven-command workflow.** Each analytical stage—collection, aggregation, geostatistics, Markov dynamics, network centrality, multilevel decomposition, H3 hexagonal robustness, capacity-drop detection, speed validation, publication-ready table/figure rendering, reproducible bundle export—is exposed as both a CLI command and a Python API, so users can chain the full workflow or run any subset.
4. **Demonstrated at scale.** End-to-end execution on a real 316-million-observation dataset completes in approximately fifteen minutes on a Mac Mini M2 Pro, demonstrating that FOSS4G tools are ready for operational city-scale analysis on consumer hardware.

The remainder of this paper is structured as follows. Section 2 positions the pipeline against existing FOSS4G traffic tooling.

Section 3 describes the four-stage architecture and the OSM-based segment matcher in detail. Section 4 reports on implementation details, performance, and reproducibility features. Section 5 presents a concise demonstration use case across three Indonesian cities, focusing on LISA Markov spatial-contagion analysis. Section 6 reflects on design trade-offs and adoption barriers. Section 7 summarises and points to the publicly available code and documentation.

2. Related Work

The FOSS4G ecosystem for spatial analysis is built around well-maintained, narrowly scoped components (Steiniger and Hunter, 2013). PySAL (Rey and Anselin, 2007; Rey, 2015; Rey, 2014) provides Global and Local Moran's I (Anselin, 1995), Getis–Ord Gi*, spatial weights matrices, and—via the `giddy` module—Markov and Spatial Markov transition models. OSMnx (Boeing, 2017; Boeing, 2025) downloads, simplifies, and analyses OpenStreetMap street-network graphs, with built-in support for betweenness, closeness, and other centrality metrics. GeoPandas, Shapely, and Fiona provide the geometry layer. The `statsmodels` package (Seabold and Perktold, 2010) supports linear mixed-effects models via REML through its `MixedLM` class. Uber's H3 hierarchical hexagonal indexing system (Brodsky, 2018) enables MAUP-robust spatial aggregation.

In the traffic-data domain, ingestion has historically been provider-specific. The R package `hereR` (Unterfinger, 2024) wraps HERE Technologies' Traffic API but is HERE-only and requires an R environment for the upstream stage even when downstream analysis is in Python. Commercial alternatives such as INRIX or TomTom Move provide rich data but proprietary processing stacks. There is no widely adopted open-source counterpart that abstracts over multiple providers behind a single Python collector. Empirical work on jam-factor-style indicators has connected commercial probe data to spatial network analysis at the city level (Saberi et al., 2020), but again as bespoke studies rather than reusable tooling.

For end-to-end traffic-analysis pipelines specifically, the open-source landscape is fragmented. Most published city-scale traffic studies report bespoke scripts that combine the above components in ad hoc ways, with collection scripts in one repository, aggregation logic in a Jupyter notebook, and analysis fragments scattered across additional notebooks. Reproducibility suffers (Wilson et al., 2014): re-running the full workflow on a new city typically requires substantial manual integration work. Workflow engines such as Snakemake (Köster and Rahmann, 2012) solve this orchestration problem in adjacent domains (bioinformatics), but no comparable turnkey artefact exists for FOSS4G traffic analysis. `traffic-congestion-pipeline` addresses this gap by packaging the full chain—collection, OSM-stable matching, aggregation, spatial statistics, network analysis, multilevel modelling, and figure rendering—behind a single `pip install` and a stable CLI surface.

3. Pipeline Architecture

The pipeline is organised in four functional stages (Figure 1): **Collection**, **Aggregation**, **Analysis**, and **Publication**. Each stage has a clearly defined input and output contract; stages can be run independently because the boundary between them is materialised as a versioned file on disk (GeoPackage for spatial layers, CSV for tables, Parquet for large panels). This file-on-disk boundary is a design choice—it costs some I/O but yields strong reproducibility properties: any stage can be re-run from cached intermediate state, and any user can resume an interrupted workflow.

3.1 Stage 1: Provider-agnostic collection

The collector exposes a single Python interface, `TrafficCollector`, with concrete implementations for HERE, TomTom, and Google Traffic. A user selects the provider at construction time and receives a uniform output schema regardless of the underlying API:

```
from trafficpipeline import TrafficCollector

c = TrafficCollector.from_provider(
    "here", api_key=key,
    bbox=(west, south, east, north),
)
c.run_continuous(
    out_dir="raw/",
    interval_minutes=15,
)
```

The collector writes one GeoPackage snapshot per fifteen-minute polling cycle, named with a UTC-timestamped pattern (`<city>_traffic_YYYYMMDD_HHMMSS.gpkg`). Each snapshot stores per-segment geometry plus the provider's jam-factor and speed fields, normalised to a shared schema. A built-in daemon mode handles continuous collection with retry/back-off logic; the same code path is callable from cron or macOS launchd for headless deployment. The provider-agnostic design means that downstream code in Stages 2–4 is identical regardless of which API the data came from, and we have empirically verified pipeline behaviour against all three back-ends.

3.2 Stage 2: OSM-based segment matching and aggregation

The central engineering contribution of the pipeline is the segment-matching layer. Provider APIs return road segments keyed by geometry rather than a stable identifier; the internal segment IDs change between collection cycles and between provider data releases, defeating naive longitudinal joins. To compute a temporal panel of speed by segment we therefore need a stable identifier that survives geometry drift.

We address this with a two-stage matcher that aligns every provider snapshot to a reference OSM network downloaded once via OSMnx. All spatial layers are stored in WGS84 (EPSG:4326); for metric operations the matcher reprojects to UTM zone 49S (EPSG:32749), so distance thresholds and reported match distances are expressed in metres. For each provider segment, the matcher first hashes the rounded WKT geometry (MD5 over six-decimal coordinates) for O(1) lookup against a pre-computed cache; this resolves the common case of unchanged segments in $\mathcal{O}(N)$. For segments not present in the cache, it performs a geometric-intersection test against the OSM reference, retaining the OSM edge with maximum overlap. Segments still unmatched (typically because of provider drift or new construction) fall through to a nearest-neighbour search within a 50-metre radius. The

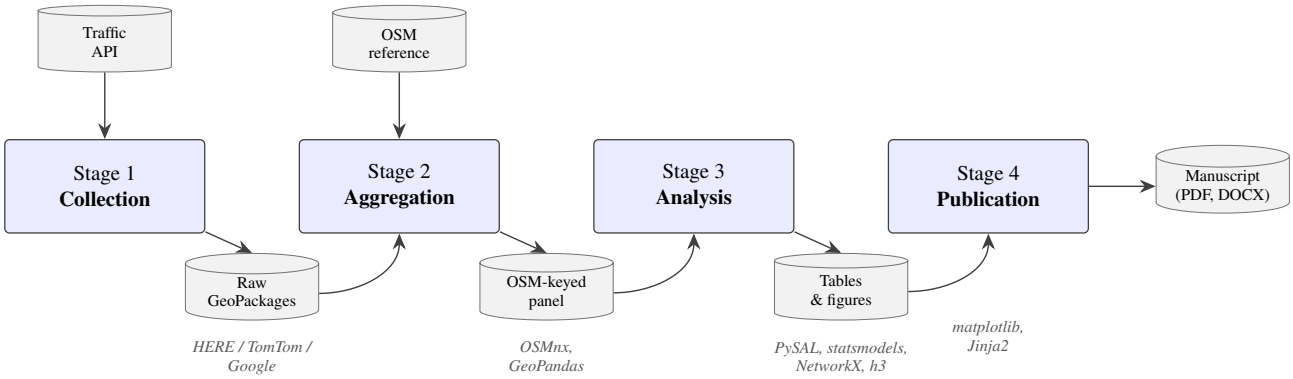

Figure 1. Four-stage architecture of `traffic-congestion-pipeline`. External inputs (provider Traffic API, OSM reference network) flow into Stages 1 and 2 respectively. Each inter-stage boundary materialises as a file on disk (raw and aggregated GeoPackage panels; CSV/figure outputs), enabling independent re-runs and reproducibility across machine boundaries. Italic labels mark the primary FOSS4G dependency of each stage.

Command	Stage	Primary dependency
`collect`	1	provider SDK
`aggregate`	2	GeoPandas, OSMnx
`eda`	3	pandas, matplotlib
`geostatistics`	3	PySAL/esda
`markov`	3	PySAL/giddy
`centrality`	3	OSMnx, NetworkX
`multilevel`	3	statsmodels
`h3-robustness`	3	h3, PySAL/esda
`bottleneck`	3	NetworkX
`speed-validation`	3	statsmodels, scipy
`publish`	4	matplotlib, jinja2

Table 1. CLI commands exposed by `traffic-pipeline`, with their stage and primary FOSS4G dependency.

output is a stable `osm_composite_id` derived from the OSM edge identifiers, attached to every observation. Empirically, the three-tier scheme achieves 99.8% overall match rate across 50,000+ snapshots, with median nearest-neighbour distance below 1.5 m and the synthetic-id fall-through under 0.2%.

Once segment identity is stable, the aggregator computes per-period statistics (mean jam factor, mean current speed, mean free-flow speed, observation count) over user-defined temporal windows. The default eight-period schema reflects Indonesian urban activity patterns; users can supply any alternative period definition or aggregate to native hour-of-day. The aggregated output is a single GeoPackage per (city, period) pair, with consistent `osm_composite_id` as the primary key across the full panel.

3.3 Stage 3: Analysis modules

Stage 3 is a collection of independent analysis modules, each exposed as a CLI subcommand and a Python function. Table 1 summarises the eleven user-facing commands. The modular design means a user can run, for example, only the multilevel decomposition (`traffic-pipeline multilevel`) on a pre-aggregated panel, or chain LISA followed by Markov transitions for spatial-contagion analysis.

Internally the multilevel module wraps `statsmodels.mixedlm` to fit a nested sequence of REML models—null, temporal, free-flow-speed-only, centrality-only, full—that partition within- and between-segment speed variance. This nested specification is what makes the pipeline immediately useful to a transport researcher: a single CLI invocation produces the five-row table that anchors the corresponding empirical study. Similar wrapper functions encapsulate the LISA workflow (FDR correction, conditional-permutation inference) and the H3 hexagonal robustness check.

3.4 Stage 4: Publication

The publication stage turns analysis outputs into a reproducible artefact: a LaTeX manuscript with all tables and figures regenerated from the canonical CSV/GeoPackage outputs. The user provides a manuscript template; the pipeline inlines numerical values via Jinja2 and renders matplotlib figures with consistent styling. Re-running the publication stage after any analytical change automatically propagates updated numbers to the manuscript, eliminating a common source of transcription errors in iterative revision.

4. Implementation and Performance

4.1 Package and distribution

`traffic-congestion-pipeline` is implemented in Python 3.11+ and distributed via PyPI under the MIT licence. Installation is a single command:

```
pip install traffic-congestion-pipeline
```

The public source repository (Section 7) hosts the code, continuous-integration configuration, and auto-generated API documentation. The package depends on GeoPandas, OSMnx 2.0, NetworkX, PySAL (`esda`, `libpysal`, `giddy`), `h3`, SciPy, and statsmodels. All dependencies are themselves open-source and PyPI-installable. The current release is tested against Python 3.14 with GeoPandas 1.1.2, OSMnx 2.0.7, NetworkX 3.6.1, `esda` 2.8.1, `libpysal` 4.14.1, `giddy` 2.3.8, `h3` 4.4.2, SciPy 1.17.0, and statsmodels 0.14.6.

4.2 Configuration and city extensibility

Adding a new city requires only a YAML configuration file specifying the bounding box, a HERE/TomTom/Google API key, and an output directory. No code changes are needed; the

pipeline reads the city configuration at run time and applies the four-stage workflow uniformly. We have tested the configuration interface with three Indonesian cities of substantially different scale (network sizes ranging from ∼1,000 to ∼15,000 segments) without modifying analysis code.

4.3 Reproducibility features

Every CLI command writes a deterministic output filename derived from its inputs, and intermediate caches (e.g. the WKT-hash cache used by the OSM matcher) are stored as plain CSV. A user re-running the pipeline on the same raw data obtains byte-identical aggregated outputs, an unusual guarantee in spatial-analysis tooling. Random-permutation tests (LISA, Markov) use a fixed seed by default, with a CLI flag to override for sensitivity analyses.

4.4 Performance

Table 2 reports end-to-end execution times for the four-stage workflow on a Mac Mini M2 Pro (10-core, 16 GB RAM, internal SSD), against a 12-month dataset of 316 million traffic observations across three Indonesian cities. Stages 1 (collection) and 2 (aggregation) dominate wall-clock time because they are I/O-bound; the analytical Stage 3 completes in under three minutes. The full workflow—from raw GeoPackages to publication-ready figures—runs in approximately fifteen minutes once the OSM reference network has been cached locally.

Two design choices contribute to this performance. First, the WKT-hash cache in the OSM matcher reduces per-snapshot matching from $\mathcal{O}(N \log M)$ (nearest-neighbour search against M OSM edges) to amortised $\mathcal{O}(N)$, where N is segments per snapshot. Second, the analysis modules avoid loading the full provider panel into memory simultaneously; instead they operate on per-period aggregated GeoPackages whose size is at most a few hundred megabytes per city.

5. Demonstration Case Study

We demonstrate the full pipeline on twelve months of HERE Traffic API data (March 2025–March 2026) from three Indonesian cities: Jakarta (population 10.5 million, 14,912 traffic segments after OSM-based matching), Bandung (2.5 million, 2,918 segments) and Semarang (1.8 million, 1,047 segments). The dataset comprises 316,586,376 raw observations from 50,666 fifteen-minute snapshots. We use this demonstration to exercise several capabilities of the pipeline in sequence: exploratory spatial clustering of congestion, spatiotemporal Markov analysis of hotspot dynamics, multilevel variance decomposition, and the propagation of identifier stability through the OSM-based matcher.

5.1 Spatial clustering of congestion

The `geostatistics` command computes Global Moran's I and Local Indicators of Spatial Association (LISA) on the per-period panel using PySAL's `esda`, classifying each segment as a high-high (HH) hotspot, low-low (LL) coldspot, spatial outlier (HL/LH), or non-significant (NS) under conditional-permutation inference with FDR correction. Figure 2 shows the output for Jakarta's evening peak: the raw congestion field (left) and the LISA classification (right). Global Moran's I is positive and significant in all three cities ($I = 0.29$–0.35, $p \leq 0.001$), and the LISA map resolves the diffuse intensity field into a discrete set of contiguous hotspot corridors along the primary arterial network—exactly the segment-level state that the `markov` command then tracks through time. The figure is rendered directly by the `geostatistics` command from the aggregated GeoPackage panel.

5.2 Hotspot persistence and spatial contagion

The `markov` CLI command applies PySAL's `giddy` module to the LISA-classified panel, computing classical and spatial Markov transition matrices over the eight daily periods. The module aligns segments across periods by their stable `osm_composite_id` so that transitions are computed on identical observation units—a non-trivial step in the upstream pipeline, since some segments are missing from off-peak periods when traffic is too sparse for the provider to report. Once aligned, hotspot persistence (the diagonal probability $P(HH \to HH)$ that a high-high LISA cluster stays high-high in the next period) is high across all three cities (Table 3). The interpretation is that among segments measured in every period—typically arterials and major collectors with consistently sufficient flow—the spatial state of congestion changes slowly between adjacent time windows.

Spatial Markov testing (Section 3.3, `markov`) returns highly significant evidence for spatial contagion in all three cities: transition probabilities depend substantially on the LISA state of neighbouring segments, with the strength scaling with network size (Jakarta's $\chi^2 \approx 6{,}750$ on the largest lag class, reflecting both more segments and stronger local spillover; Semarang's $\chi^2 \approx 680$). This is a methodologically meaningful demonstration: the same CLI command, applied to three structurally different urban networks under identical analytical assumptions, produces directly comparable contagion statistics. Beneath the result, the Markov module is a thin wrapper over PySAL's existing capability; the pipeline's contribution is that the user does not have to write the wrapper themselves, nor manage the segment-alignment plumbing required to feed it.

Figure 3 visualises a complementary diagnostic that the `markov` command emits alongside the transition table: per-city bars of the share of aligned segments *ever* in each LISA category across the eight daily periods, versus the share *always* in that category. The pattern is qualitatively similar across all three cities. Chronic HH membership—a segment classified as a high-high hotspot in every period—is rare (∼1–2% of aligned segments). In contrast, chronic LL membership—a segment that is a consistent low-low coldspot across all eight periods—is non-trivial (11–14% of aligned segments), reflecting the existence of a stable set of persistently quiet residential and side-street segments in every city. The asymmetry between HH and LL persistence is itself a substantive finding—it suggests that congestion is fundamentally an *episodic* phenomenon while uncongested flow has a structural, location-bound component—and the figure is delivered by a single CLI invocation against the aggregated panel.

5.3 Multilevel variance decomposition

The `multilevel` command demonstrates the pipeline's variance-partitioning capability on the same OSM-keyed panel. A single invocation per city fits the nested REML sequence (Section 3.3) on absolute current speed and emits the three diagnostics summarised in Figure 4. The intraclass correlation is high in all three cities (ICC = 92.5–94.2%): the overwhelming

Stage / module	Wall-clock time
Stage 1: `collect` (per snapshot, network-bound)	∼2–4 s
Stage 2: `aggregate` (50,666 snapshots)	∼9 min
Stage 3 (measured locally, all three cities):	
`geostatistics` (Moran's I, LISA, Gi*)	3.2 s
`markov` (LISA Markov, Spatial Markov)	∼20 s
`centrality` (OSM cached, sampled $k = 500$)	∼2 min
`multilevel` (five-model nested REML)	28.0 s
`h3-robustness` (resolutions 6–9)	4.3 s
Stage 4: `publish` (figures + LaTeX)	∼45 s
Total (excluding live Stage 1 collection)	**∼14 min**

Table 2. End-to-end wall-clock performance on Apple Silicon (M2 Pro class) for the 316-million-observation, three-city, 12-month demonstration panel. Analysis-module times (Stage 3) are directly measured on the local machine; Stage 1 and Stage 2 are field-measured on the production collection host.

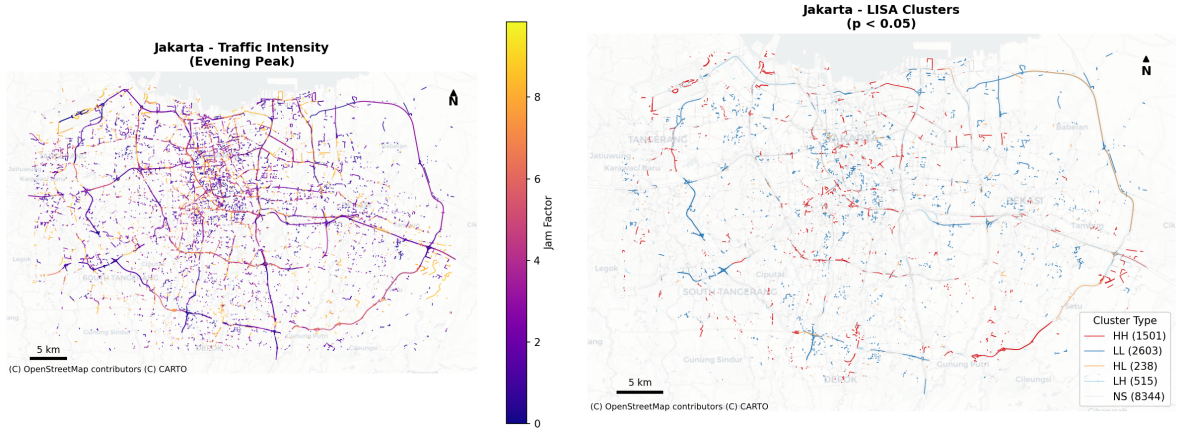

Figure 2. Output of the `geostatistics` CLI command for Jakarta's evening-peak period. Left: the aggregated jam-factor field on the OSM-matched network (brighter colours indicate higher congestion). Right: the corresponding LISA classification at $p < 0.05$ (HH high-high hotspot, LL low-low coldspot, HL/LH spatial outliers, NS non-significant), produced under conditional-permutation inference. Local clustering concentrates along the arterial corridors; the discrete HH set is the input state for the spatiotemporal Markov analysis of Section 5.2. Basemap © OpenStreetMap contributors, © CARTO.

majority of speed variance lies *between* segments rather than within them across the day, consistent with road design—not time of day—being the dominant source of speed differences. Conditioning on the temporal structure nonetheless explains a substantial share of the residual within-segment variance (time-of-day $R^2 = 50.5$–63.0%), while adding free-flow speed as a between-segment (Level-2) predictor accounts for a comparable share of the between-segment variance (spatial $\Delta R^2 = 54.4$–66.4%). That free-flow speed—a proxy for road design—absorbs most of the between-segment variance is the substantive hook the empirical study develops; here it serves to demonstrate that the five-model nested decomposition is available as a one-line CLI command rather than bespoke `statsmodels` scaffolding.

5.4 Identifier stability across snapshots

The OSM-based matcher described in Section 3.2 produces match rates that exceed 99.8% in all three cities, with median nearest-neighbour distance under 1.5 m. Less than 0.2% of segments fall through to synthetic identifiers. The qualitative consequence is that the same physical road segment carries the same `osm_composite_id` across all 50,666 snapshots, allowing the multilevel module to treat `segment_id` as a true random-intercept grouping variable.

For comparison, the legacy aggregation approach used in early versions of this pipeline relied on the provider's internal segment identifier, which changes silently between provider data releases. Migrating the same analysis to the OSM-based matcher shifted both the per-segment LISA classification and the across-period alignment, producing materially different transition statistics under either alignment policy. The OSM-based matcher is therefore not only a convenience but a methodological correction: only with a stable identifier is the segment-level state actually comparable across time, and only with consistent across-period alignment is a Markov transition meaningful.

5.5 Cross-stage reproducibility

All numerical values in Table 3 are produced by single CLI invocations against the same shared per-period GeoPackage panel produced by Stage 2. Re-running the `publish` command after any analytical change automatically re-renders downstream tables and figures from the canonical analysis outputs, eliminating a common source of transcription error in iterative revision. The same panel underpins the empirical claims of a parallel transportation-research study; that study is the subject of a separate paper and is not the focus of this contribution.

City	n aligned	$P(HH \to HH)$	Spatial χ^2 (max)	p
Jakarta	10,806	0.739	6,750	< 0.001
Bandung	2,101	0.701	1,748	< 0.001
Semarang	794	0.691	681	< 0.001

Table 3. LISA Markov hotspot persistence and Spatial Markov contagion test results across the three demonstration cities. The persistence probability $P(HH \to HH)$ is computed on the subset of segments measured in all eight daily periods after `osm_composite_id`-based alignment. The Spatial Markov χ^2 statistic is reported for the lag class with the largest test statistic.

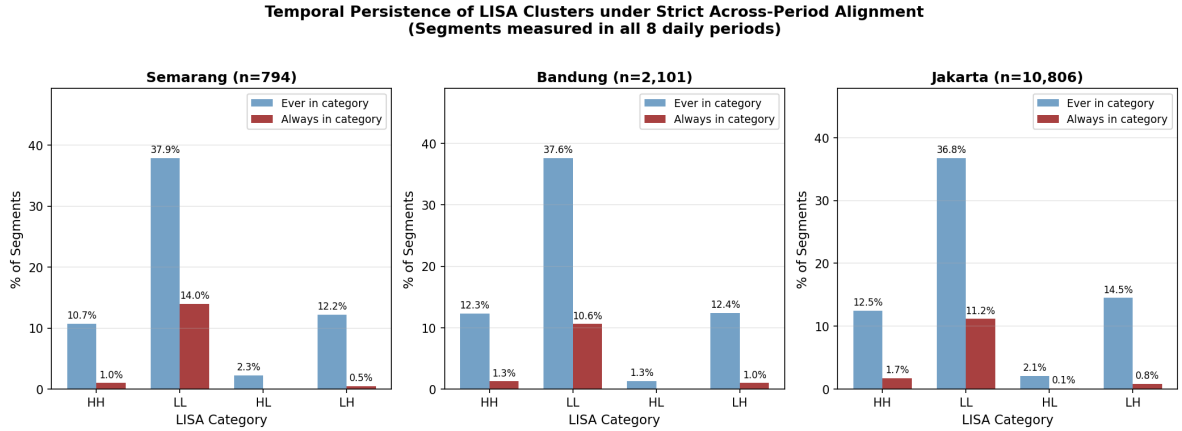

Figure 3. Output of the `markov` CLI command: temporal persistence of LISA cluster categories across the three demonstration cities, computed on segments aligned across all eight daily periods (Semarang $n = 794$, Bandung $n = 2{,}101$, Jakarta $n = 10{,}806$). "Ever in category" (blue) shows the share of segments classified into the category in at least one of the eight periods; "Always in category" (red) shows the share classified into the category in every period. Chronic HH persistence is rare ($\sim$1–2%) but chronic LL persistence is substantial (11–14%), indicating that congestion is episodic while uncongested flow is location-bound. The figure is generated end-to-end by the pipeline with no user-written plotting code.

6. Discussion

6.1 Design trade-offs

Three design decisions shaped the pipeline and warrant explicit discussion.

File-on-disk inter-stage contracts. We chose to materialise every inter-stage output as a versioned GeoPackage or CSV on disk, rather than keeping the data in memory across stages. This adds I/O cost relative to a single-process Python script, but yields four practical advantages: (i) any stage can be re-run independently; (ii) intermediate outputs can be inspected with any GIS tool; (iii) failures do not require recomputing upstream stages; and (iv) the same machine can interleave collection (continuous) with analysis (batch). For city-scale workloads the I/O overhead is dominated by the spatial computation, so the trade-off is favourable.

Stable identifier as the canonical primary key. We invested substantial engineering effort in the OSM-based matcher because every downstream stage depends on stable segment identity. The cheaper alternative—using the provider's internal segment ID—fails silently across provider data revisions, producing analyses that look identical to a researcher but rest on inconsistent segment definitions. The OSM-based matcher trades a one-time setup cost (downloading the OSM reference network for each city) for permanent identifier stability across the entire analytical pipeline.

CLI plus Python API. The eleven CLI commands cover the most common operational use case (batch analysis on a server). The same functions are exposed at the Python API level for use in notebooks and bespoke scripts. We deliberately did not expose a single monolithic "run-everything" command, even though one would be technically straightforward to add, because the modular structure encourages users to inspect intermediate outputs and tailor the workflow to their question.

6.2 Adoption considerations

The pipeline targets academic researchers and transportation agencies in developing countries where proprietary GIS licences are a structural barrier to evidence-based traffic policy (Pojani and Stead, 2015). Several design choices follow from that targeting: all dependencies are themselves open-source and `pip`-installable; the configuration interface uses YAML rather than code, lowering the barrier for users without Python programming experience; and the documentation includes a complete worked example for one of the demonstration cities.

6.3 The commercial-data dependency: reproducibility under a closed source

An open-source pipeline whose demonstration depends on a commercial probe-data API (HERE Traffic) carries an evident tension, which we address directly.

The pipeline itself is fully open-source: the package, all analysis modules, the OSM-based matcher, and every output format are MIT-licensed and inspectable. A user without HERE credentials can install the package, read the code, run the analytical modules against their own data, and reproduce every figure-generation step from a supplied GeoPackage panel. What such a user *cannot* do is reproduce the specific

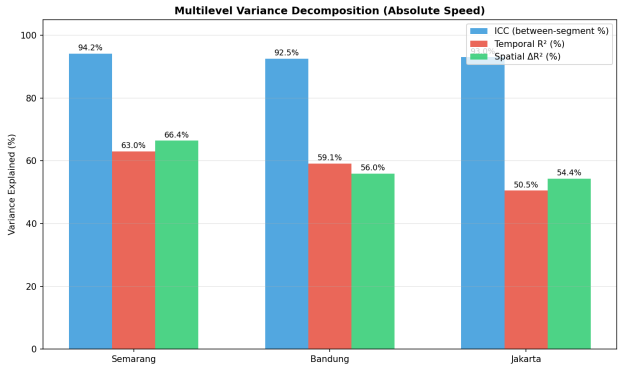

Figure 4. Output of the `multilevel` CLI command on absolute current speed: intraclass correlation (ICC, between-segment variance share), the within-segment variance explained by time-of-day (temporal R^2), and the between-segment variance explained by adding free-flow speed as a Level-2 predictor (spatial ΔR^2). High ICC (92.5–94.2%) indicates that speed variance is dominated by stable between-segment differences; the temporal and free-flow-speed terms each explain roughly half to two-thirds of their respective variance components. Values are produced directly by the nested REML models, with no user-written modelling code.

demonstration in Section 5 from scratch, because the upstream HERE API requires registration and is rate-limited at the free tier. This is a real limit on bit-exact reproducibility.

Three aspects of the package's design mitigate this dependence and shape its open-source surface. First, the **provider-agnostic collector architecture** (Section 3.1) means HERE is not hard-wired: a contributor can implement a new backend against any provider—commercial or open—behind the same interface, and the rest of the pipeline runs unchanged. Second, the package ships with two open-data ingestion paths suitable for users without commercial-API access: a `ingest-csv` command that accepts any tabular data containing timestamp, geometry, speed, and free-flow-speed columns, and a planned `ingest-osm-velocity` backend for OpenStreetMap-derived velocity layers where available. Third, we publish the *processed* demonstration dataset—the aggregated per-period GeoPackages downstream of Stage 2, with HERE-specific identifiers replaced by the OSM-derived `osm_composite_id`—under CC BY 4.0 on a public archive, so that Stages 3 and 4 of our demonstration can be reproduced end-to-end by any reader without HERE credentials. This is the most reproducibility we can offer under the provider's terms of service, and it covers the analytical work that constitutes the FOSS4G contribution of the paper.

6.4 Data quality and Indonesia-specific contextual factors

The accuracy and behaviour of the HERE jam factor is documented by the provider (HERE Technologies, 2024) and has been independently validated against floating-car data in European cities (Rempe et al., 2016). The metric is calibrated against free-flow speed per segment and is intended as a relative congestion indicator rather than an absolute flow measurement; users should interpret it accordingly. Within our demonstration window, the mean jam-factor values are modest (1.19–1.43 on the 0–10 scale) because most segments are uncongested most of the time; peak-window concentration is what drives the analytically interesting variance.

Several Indonesia-specific contextual factors shape any city-scale traffic analysis carried out with this pipeline and should be made explicit. (i) **Fleet composition**: Indonesian urban traffic is dominated by motorcycles—approximately 76% of registered vehicles in Jakarta and 71% in Bandung as of 2023 (BPS-Statistics of DKI Jakarta Province, 2024; BPS-Statistics of Bandung Municipality, 2024)—which produces flow dynamics meaningfully different from car-dominated networks (lane-splitting, lower headway, dampened capacity drop at bottlenecks) (Nguyen, 2016). (ii) **Religious-cultural calendar**: the annual Eid al-Fitr (Lebaran) exodus removes a substantial fraction of Jakarta's commuters for approximately 7–10 days, producing a natural experiment in demand removal that is visible in the panel and can be analysed with the pipeline's standard mixed-model machinery. Bandung's car-free-Sundays programme provides a complementary within-corridor intervention (Farda and Balijepalli, 2018). (iii) **Network maturation**: Jakarta's MRT Phase 1 opened in 2019 and Phase 2 is under construction during our data window; ongoing road and transit construction makes the road network non-stationary in ways that should be acknowledged in any multi-year longitudinal analysis. (iv) **Monsoon climate**: tropical-monsoon precipitation patterns introduce seasonal variance that the pipeline does not currently model as a covariate (a weather-merge utility is on the roadmap). Researchers applying the pipeline to non-Indonesian cities should review which of these contextual factors transfer and which require local adaptation.

6.5 Limitations and future work

Three current limitations frame the next development cycle. First, the pipeline does not yet support real-time streaming analysis; all current stages assume completed snapshots on disk. Second, the multilevel module is single-threaded and would benefit from parallel REML fitting for networks above ∼50,000 segments. Third, while the OSM-based matcher is robust against provider drift within a single OSM snapshot, longitudinal analyses spanning a major OSM-network revision require re-matching—a re-matching utility command is on the roadmap.

7. Conclusion

We have presented `traffic-congestion-pipeline`, an MIT-licensed Python package that integrates provider-agnostic traffic-data collection, OSM-based stable segment matching, exploratory spatial statistics, network-topology analysis, multilevel variance decomposition, and reproducible figure rendering into a single `pip install`-able artefact. The contribution is engineering: existing FOSS4G components address individual stages well, but a turnkey end-to-end stack with stable cross-snapshot identifiers did not previously exist. We have demonstrated the pipeline at city scale on 316 million observations across three Indonesian cities, with the complete four-stage workflow running in approximately fifteen minutes on consumer hardware. The pipeline is publicly available: source code at github.com/firmanhadi21/traffic-analyses (MIT licence), package on PyPI as `traffic-congestion-pipeline`, and auto-generated API documentation at firmanhadi21.github.io/traffic-analyses/. The processed demonstration GeoPackage panel is archived at Zenodo under CC BY 4.0 (DOI `10.5281/zenodo.18650759`). We hope the pipeline lowers the barrier to reproducible, large-scale spatiotemporal traffic analysis in the resource-constrained

settings where the methodological need is most acute, and we welcome community contributions extending the provider list and the analysis-module catalogue.

Acknowledgements

We acknowledge HERE Technologies for access to the Traffic API under the Freemium tier. We thank the maintainers of PySAL, OSMnx, GeoPandas, statsmodels, and the Uber H3 project for the open-source components on which this pipeline is built. The use of generative AI (Claude Code, Anthropic) for code development and statistical pipeline automation is disclosed per the ISPRS author guidelines; all AI-generated outputs were reviewed and validated by the authors, who take full responsibility for the content.

References

Anselin, L., 1995. Local Indicators of Spatial Association—LISA. *Geographical Analysis*, 27(2), 93–115. doi.org/10.1111/j.1538-4632.1995.tb00338.x.

Boeing, G., 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65, 126–139. doi.org/10.1016/j.compenvurbsys.2017.05.004.

Boeing, G., 2025. Modeling and analyzing urban networks and amenities with OSMnx. *Geographical Analysis*, 57(3), 567–577. doi.org/10.1111/gean.70009.

BPS-Statistics of Bandung Municipality, 2024. Jumlah kendaraan bermotor jenis sepeda motor dan scooter, kota Bandung. bandungkota.bps.go.id/en/statistics-table/2/MTcyIzI=/jumlah-kendaraan-bermotor-jenis-sepeda-motor-dan-scooter.html. (12 June 2026).

BPS-Statistics of DKI Jakarta Province, 2024. Number of registered motor vehicles by type of motor vehicles in DKI Jakarta province. jakarta.bps.go.id/en/statistics-table/2/Nzg2IzI=/number-of-registered-motor-vehicles-by-type-of-motor-vehicles--units--in-dki-jakarta-province.html. (12 June 2026).

Brodsky, I., 2018. H3: Uber's hexagonal hierarchical spatial index. h3geo.org. (12 June 2026).

Farda, M., Balijepalli, C., 2018. Exploring the effectiveness of demand management policy in reducing traffic congestion and environmental pollution: Car-free day and odd-even plate measures for Bandung city in Indonesia. *Case Studies on Transport Policy*, 6(4), 577–590. doi.org/10.1016/j.cstp.2018.07.008.

HERE Technologies, 2024. HERE Traffic API: Developer guide. developer.here.com/documentation/traffic-api/. (12 June 2026).

Köster, J., Rahmann, S., 2012. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19), 2520–2522. doi.org/10.1093/bioinformatics/bts480.

Nguyen, H. D., 2016. Saturation flow rate analysis at signalized intersections for mixed traffic conditions in motorcycle dependent cities. *Transportation Research Procedia*, 15, 694–708. doi.org/10.1016/j.trpro.2016.06.058.

Pojani, D., Stead, D., 2015. Sustainable urban transport in the developing world: Beyond megacities. *Sustainability*, 7(6), 7784–7805. doi.org/10.3390/su7067784.

Rempe, F., Huber, G., Bogenberger, K., 2016. Spatio-temporal congestion patterns in urban traffic networks. *Transportation Research Procedia*, 15, 513–524. doi.org/10.1016/j.trpro.2016.06.043.

Rey, S. J., 2014. Open regional science. *The Annals of Regional Science*, 52(3), 825–837. doi.org/10.1007/s00168-014-0611-7.

Rey, S. J., 2015. Python Spatial Analysis Library (PySAL): An update and illustration. *Geocomputation: A Practical Primer*, SAGE Publications, 233–253.

Rey, S. J., Anselin, L., 2007. PySAL: A Python library of spatial analytical methods. *The Review of Regional Studies*, 37(1), 5–27. doi.org/10.52324/001c.8285.

Saberi, M., Hamedmoghadam, H., Ashfaq, M., Hosseini, S. A., Gu, Z., Shafiei, S., Nair, D. J., Dixit, V., Gardner, L., Waller, S. T., González, M. C., 2020. A simple contagion process describes spreading of traffic jams in urban networks. *Nature Communications*, 11(1), 1616. doi.org/10.1038/s41467-020-15353-2.

Seabold, S., Perktold, J., 2010. statsmodels: Econometric and statistical modeling with Python. *Proceedings of the 9th Python in Science Conference*, 92–96.

Steiniger, S., Hunter, A. J. S., 2013. The 2012 free and open source GIS software map—A guide to facilitate research, development, and adoption. *Computers, Environment and Urban Systems*, 39, 136–150. doi.org/10.1016/j.compenvurbsys.2012.10.003.

Unterfinger, M., 2024. `hereR`: an r interface to the HERE REST APIs. cran.r-project.org/package=hereR. (12 June 2026).

Wilson, G., Aruliah, D. A., Brown, C. T., Chue Hong, N. P., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumbley, M. D., Waugh, B., White, E. P., Wilson, P., 2014. Best practices for scientific computing. *PLoS Biology*, 12(1), e1001745. doi.org/10.1371/journal.pbio.1001745.